



Машинное обучение для решения исследовательских и инженерных задач в науках о Земле

Михаил Криницкий

к.т.н., н.с.

Институт океанологии РАН им. П.П. Ширшова

Лаборатория взаимодействия океана и атмосферы и
мониторинга климатических изменений (ЛВОАМКИ)



Непараметрические методы машинного обучения

Михаил Криницкий

к.т.н., н.с.

Институт океанологии РАН им. П.П. Ширшова

Лаборатория взаимодействия океана и атмосферы и
мониторинга климатических изменений (ЛВОАМКИ)

Деревья решений



Деревья решений

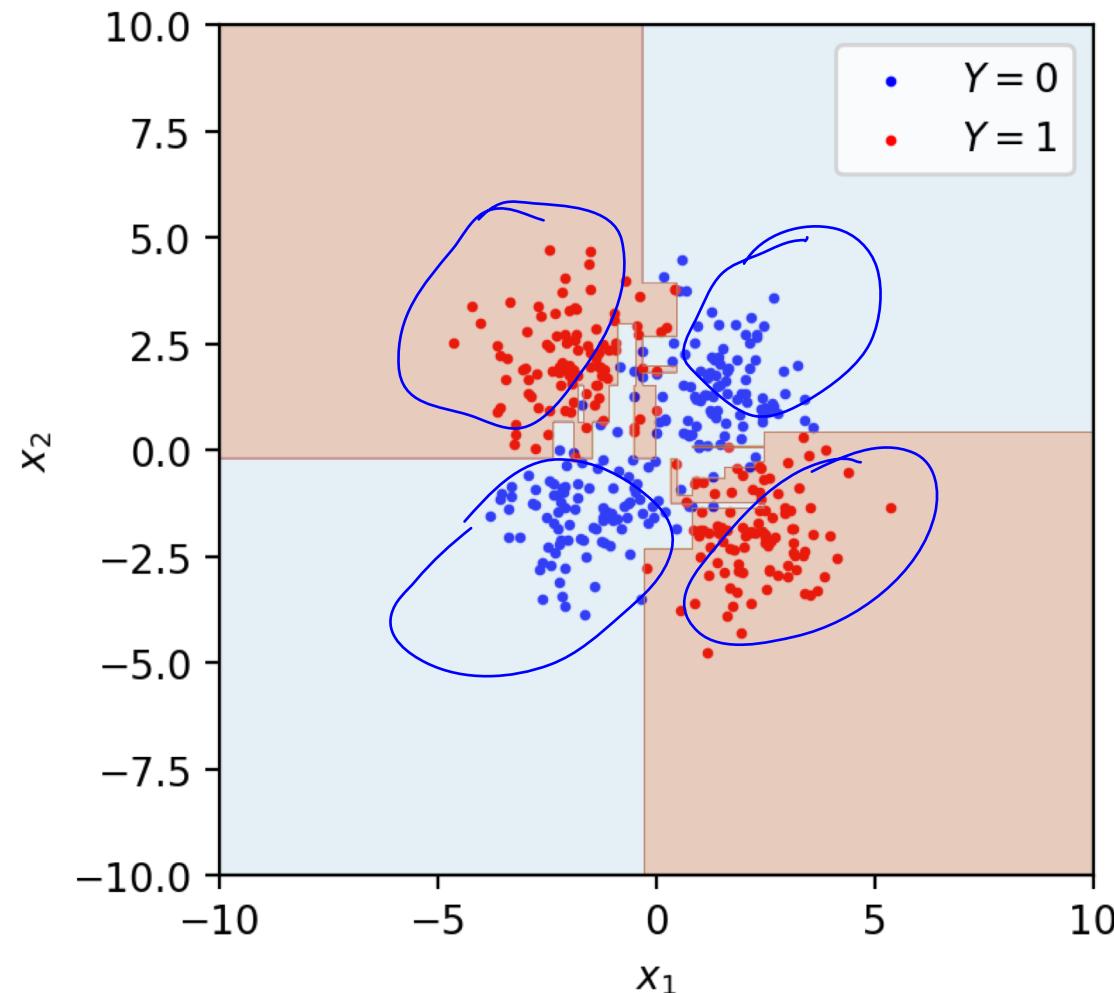
Особенности метода

- деревья решений **сильно склонны к переобучению**; для борьбы с этим можно применять эвристические способы ограничения выразительной способности во время обучения:
 - ограничивать глубину дерева (`max_depth`*): если достигнута максимальная глубина, критерий останова ветвления считается выполненным;
 - ограничивать минимальный размер листа (`min_samples_leaf`*): если количество элементов в очередном листе не превышает этого значения, критерий останова ветвления в этой ветке считается выполненным;
 - ограничивать максимальное количество листьев дерева при его построении (`max_leaf_nodes`*): если количество листьев достигло определенного предела, обучение всего дерева прекращается;
 - ограничивать минимальное снижение функции потерь при ветвлении (`min_impurity_decrease`*): если максимально достижимое снижение функции потерь при ветвлении не превышает порогового, критерий останова ветвления в этой ветке считается выполненным

Эти гиперпараметры можно оптимизировать на основании меры качества на валидационной выборке, на выборках ОOB в подходе `bootstrap` или в подходе скользящего контроля.

- альтернативно: ведется полное обучение всего дерева без ограничений, после чего выполняется обрезка (pruning) дерева: оставляются ветвления до определенной глубины (подбирается на основании меры качества на валидационной выборке).

НИКОГДА НЕ ПРИМЕНЯЙТЕ деревья решений как таковые!



Ансамбли моделей

деревья решений **сильно склонны к переобучению.** **НИКОГДА НЕ ПРИМЕНЯЙТЕ** деревья решений как таковые!

Способ борьбы с этой особенностью – использование ансамблей алгоритмов.



“Ансамбль искусственных нейронов” by Alexander Gavrikov

Ансамбли моделей

деревья решений **сильно склонны к переобучению**. **НИКОГДА НЕ ПРИМЕНЯЙТЕ** деревья решений как таковые!

Способ борьбы с этой особенностью – ансамблирование моделей.

Виды ансамблей:

- **Weighted averaging** (взвешенное осреднение): обучить K различных методов («базовых алгоритмов») на одних и тех же данных; результат взвешенно осреднять (в случае регрессии) или получать взвешенным голосованием (в случае классификации):

$$\hat{y}_i = \frac{1}{K} \sum_{k=1}^K w_k y_i^{(k)}$$
$$\hat{c}_i = \operatorname{argmax}_{c \in \mathbb{Y}} \sum_{k=1}^K w_k * [c_i^{(k)} == c]$$

$|L=8$

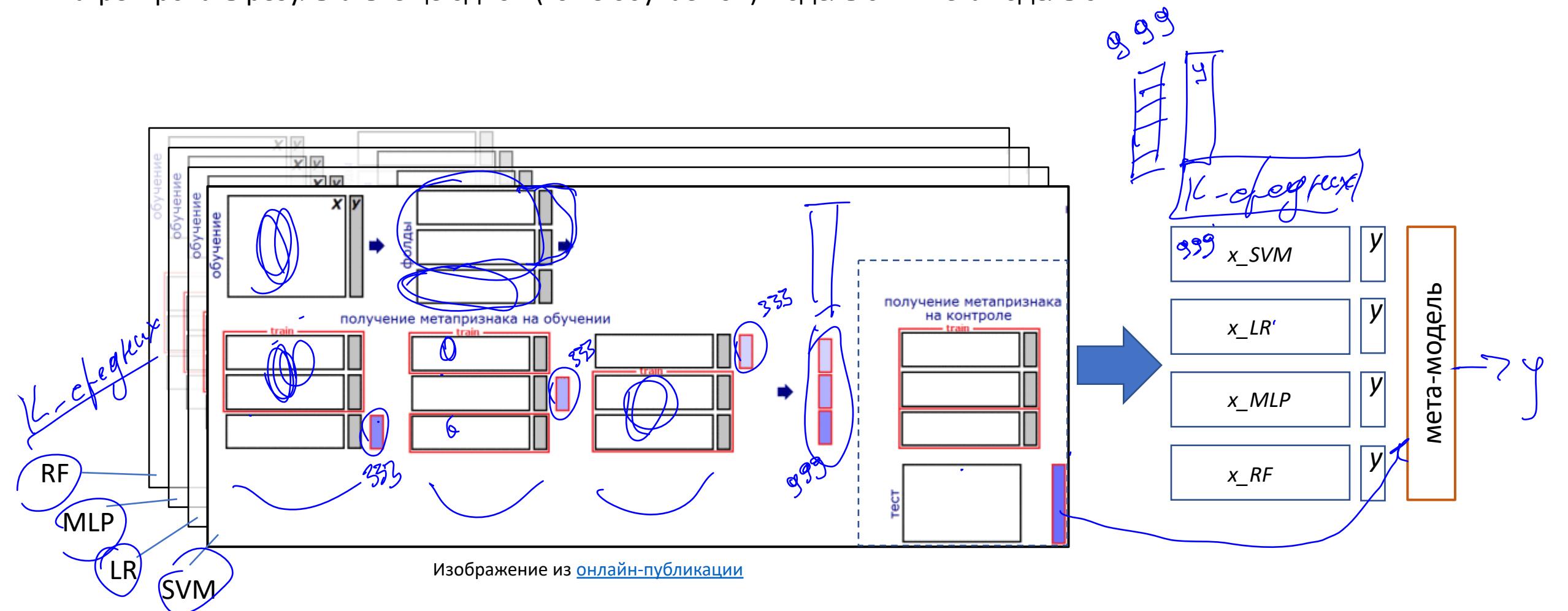
если все веса w_k равны 1, получим простое голосование/осреднение; в случае, когда веса зависят от x (а значит обучаемые) – получим т.н. «смесь экспертов» (blending)

- **Stacking** (стекинг): обучить К различных базовых алгоритмов на одних и тех же данных; вывод каждого из алгоритмов (значения параметров целевого распределения μ_i или p_i) использовать как новые признаки для новой мета-модели (обычно довольно простого, напр., любой из вариантов GLM/GAM: линейная регрессия в случае регрессии или логистическая регрессия в случае классификации);
- Обучать один и тот же базовый алгоритм на K полностью различных тренировочных выборках; результаты взвешенно осреднять (см. выше) или использовать эти K моделей в подходе стекинга. При этом рассчитывать на то, что каждой из этих выборок достаточно для обучения модели; все они порождены из одного и того же распределения. Однако набирать две или больше достаточно объемные тренировочные выборки – дорого и долго;
- **Random Subspace Method** (метод случайных подпространств): обучать К базовых алгоритмов (различных или одинаковых)
- **Bagging** (Bootstrap Aggregating, агрегирование в подходе бутстрэп) – см. далее;
- **Boosting** («бустинг») – см. далее.

Ансамбли моделей: stacking

Идея:

1. обучить множество нескольких базовых алгоритмов, каждый из которых где-то хорошо работает, а где-то систематически ошибается, получать этими моделями т.н. «метапризнаки»;
2. агрегировать результаты еще одной (тоже обучаемой) моделью – «метамоделью».



Ансамбли моделей: bagging

Bagging (Bootstrap Aggregating, агрегирование в подходе бутстрэп)

Идея:

1. обучить множество базовых алгоритмов, склонных к переобучению, на подвыборках, гарантированно порожденных одним и тем же распределением (identically distributed, "i.d.");
2. агрегировать результаты в подходе простого голосования/осреднения.

Сэмплирование из тренировочной выборки в подходе Bootstrap гарантирует* идентичность порождающего распределения**. В случае ограниченного количества выборок Bootstrap предоставляет лучшее из доступных приближений***.

Размер каждой выборки bootstrap:

- в случае сильно ограниченного размера тренировочной выборки – берут размером с тренировочную;
- в случае большого тренировочного набора данных размер bootstrap-выборки – гиперпараметр, подбирается по качеству на валидационной выборке.

Почему вообще ансамблирование одинаковых переобучающихся алгоритмов может работать, если сам алгоритм «плохой»?

* в пределе бесконечного количества выборок

** в смысле статистик, оцениваемых эмпирически

*** Efron B. Bootstrap Methods: Another Look at the Jackknife Springer Series in Statistics / под ред. S. Kotz, N.L. Johnson, New York, NY: Springer, 1992. 569–593 с.

Ансамбли моделей: bagging

$K = 500$

Bagging (Bootstrap Aggregating, агрегирование в подходе бутстрэп)

Почему вообще ансамблирование K одинаковых переобучающихся алгоритмов может работать, если сам алгоритм «плохой»?

Оценка целевой переменной (точнее, какого-то параметра распределения $P(y|x)$, например, $\mu(x)$) – тоже случайная величина (обозначим T), с определенным распределением $P(T)$.

Обычно (в предположении, что T_1, T_2, \dots, T_K – оценки K разными алгоритмами, i.i.d.* случайные величины):

$$\overbrace{Var(T_1)}^{\text{---}} = \overbrace{Var(T_2)}^{\text{---}} = \cdots = \overbrace{Var(T_n)}^{\text{---}} = \sigma^2$$

тогда

$$Var(\bar{T}) = Var\left(\frac{1}{K} \sum_{k=1}^K T_k\right) = \frac{\sigma^2}{K}$$

Представим, что эти случайные величины – не независимы (в случае обучения K одинаковых алгоритмов на bootstrap-выборках уже нельзя говорить о независимости результатов). Например (простейший вариант), попарные корреляции между ними одинаковы и составляют ρ :

$$\rho = \frac{Cov(T_j, T_i)}{\sigma_{T_i} \sigma_{T_j}}$$
$$Cov(T_i, T_i) = Var(T_i) = \sigma^2$$

Тогда:

$$Var(\bar{T}) = Var\left(\frac{1}{K} \sum_k T_k\right) = \frac{1}{K^2} \sum_{i,j} Cov(T_i, T_j) = K \frac{\sigma^2}{K^2} + \frac{K(K-1)}{K^2} \rho \sigma^2 = \boxed{\rho \sigma^2 + \frac{1-\rho}{K} \sigma^2}$$

*i.i.d. – independent, identically distributed

Ансамбли моделей: bagging

Bagging (Bootstrap Aggregating, агрегирование в подходе бутстрэп)

$K=500$

$$Var(\bar{T}) = \rho\sigma^2 + \frac{1 - \rho}{K}\sigma^2$$

где T – случайная переменная оценки параметра условного распределения $P(y|x)$ целевой переменной (μ для регрессии, ρ для классификации)
 $Var(\bar{T})$ - дисперсия средней оценки этого параметра при ансамблировании K одинаковых алгоритмов при смягчении предположения о независимости их ответов (например, при обучении на пересекающихся bootstrap-выборках)

Выводы: для снижения дисперсии (неопределенности) ответов ансамбля

- следует повышать K – количество членов ансамбля
- следует снижать ρ , характеризующую степень их коррелированности – делать результаты базовых алгоритмов как можно менее похожими

Bagging эксплуатирует подход обучения большого количества ($K \gg 1$) моделей, склонных к переобучению (σ^2 – существенна, но ρ сильно меньше единицы, алгоритмы раскоррелированы за счет склонности к переобучению и за счет обучения на различающихся подвыборках).

Способ применения в случае решающих деревьев: обучить очень много довольно решающих деревьев до конца (не ограничивая их глубину, без регуляризаций); обучать на bootstrap-выборках, агрегировать результаты по принципу простого голосования (в случае классификации) или простого осреднения (в случае регрессии).

Ансамбли моделей: Random Forests

Bagging + Random Subspace Method =^{*} Random Forests^{**}

$$Var(\bar{T}) = \rho\sigma^2 + \frac{1 - \rho}{K}\sigma^2$$

Идея метода Случайных Лесов: снизить корреляцию ρ результатов базовых алгоритмов еще больше (по сравнению с подходом бэггинга над решающими деревьями) за счет выбора случайных подпространств (совокупности признаков), на которых ищется оптимальное ветвление на итерациях обучения решающих деревьев.

Псевдоалгоритм обучения случайных лесов:

Начальное состояние:

Выборка $R = \{x_i, y_i\}$; множество признаков $x_i \in F$; кол-во деревьев в композиции B (задается исследователем); множество базовых алгоритмов H – пустое.

ФУНКЦИЯ RANDOM_FOREST(R):

Повторять B раз:

1. $R_k = bootstrap(R)$
2. $h_k = RANDOMIZED_TREE(R_k)$
3. $H = H \cup h_k$

Возврат H

ФУНКЦИЯ RANDOMIZED_TREE(R_k):

В каждом узле ветвления:

1. f – небольшое подмножество признаков F
2. поиск оптимального разделения $s(j_l, t^{(l)})$ только среди признаков из подмножества f

Возврат обученного дерева



"Случайный лес" by Alexander Gavrikov

* Не совсем так. См. псевдоалгоритм случайных лесов

** Breiman L. Random Forests // Machine Learning. 2001. № 1 (45). С. 5–32.

Ансамбли моделей: Bagging, Random Forests

$$Var(\bar{T}) = \rho\sigma^2 + \frac{1 - \rho}{K}\sigma^2$$

Особенности бэггинга (включая RF):

- + Возможность снижения дисперсии оценки параметра распределения целевой переменной, применяя слабые базовые алгоритмы (для RF это свойство выражено еще сильнее)
- + => снижение неопределенности решения
- + => повышение точности решения
- + «бесплатная» валидация – оценка качества на ОOB-выборках, получаемых при bootstrap-сэмплировании
- + сниженная чувствительность к выбросам и отсутствующим значениям (за счет применения метода случайных подпространств, за счет bootstrap-сэмплирования)
- + повышение количества членов ансамбля не приводит к переобучению, зато приводит к снижению дисперсии ответов
- слишком низкая выразительная способность членов ансамбля может приводить к низкой выразительной способности всей композиции
- решение ансамбля сложнее интерпретировать (по сравнению с GLM/GAM или DT)
- более вычислительно затратны при обучении по сравнению с GLM/GAM или DT

Ансамбли моделей: boosting

$K = 100$

Идея:

Обучать слабые базовые алгоритмы (“weak classifier/regressor”) с низкой выразительной способностью – последовательно, с итерационным изменением весов w_i примеров x_i тренировочной выборки. Веса примеров модифицировать, руководствуясь ошибками композиции, построенной к этой итерации.

Применение в случае решающих деревьев: отдельные деревья обучаются сильно регуляризую, например, сильно ограничивая глубину (в случае глубины в 1-2 ветвления они называются «решающими пнями», decision stumps)

AdaBoost:

Начальное состояние:

Выборка $R = \{x_i, y_i\}$, количество примеров $N = |R|$; ансамбль $H(x)$ – константная модель с предустановленным решением для любого примера. Начальные значения весов примеров: $w_i = 1/N$ для всех $i = 1 \dots N$. Количество членов ансамбля K (задается исследователем).

Повторять K раз, $k = 1 \dots K$:

1. Создать и оптимизировать слабый базовый алгоритм g_k на обучающей выборке с учетом весов примеров $\{w_i\}$
2. Вычислить взвешенную ошибку этого классификатора: $err_k = \frac{1}{\sum w_i} \sum w_i * [g_k(x_i) \neq y_i]$
3. Вычислить фактор модификации весов: $\alpha_k = \ln \frac{1 - err_k}{err_k}$
4. Адаптировать веса примеров, на которых допущена ошибка: $w_i = w_i * \exp(\alpha_k [g_k(x_i) \neq y_i])$

Итоговый алгоритм – агрегирующая композиция всех обученных слабых алгоритмов с соответствующими весами:

$$F(x) = \sum_k \alpha_k g_k(x)$$

Ансамбли моделей: gradient boosting*

Идея:

- в подходе бустинга воспринимать построение композиции как задачу градиентной оптимизации в отношении некоторой функции потерь в пространстве функций (базовых алгоритмов):

$$d_k(x_i) = \frac{\partial \mathcal{L}(y, F_k(x_i))}{\partial F_k(x_i)}$$

- на каждом шаге градиентной оптимизации обучается новый слабый алгоритм, аппроксимирующий $g_k(x_i)$ с той точностью, с которой позволяет его выразительная способность:

$$\tilde{d}_k(x_i) = \operatorname{argmin}_{\gamma} \sum_{i=1}^N (g(x_i, \gamma) - d_k(x_i))^2$$

(в случае функции ошибки MSE в задаче регрессии)

- в отношении композиции производится итерация градиентной оптимизации с шагом η :

$$F_{k+1}(x) = F_k(x) + \beta \tilde{d}_k(x)$$

Подход градиентного бустинга также называют подходом пошагового аддитивного моделирования (Stepwise Additive Modeling)

* Breiman L. Technical Report 486, Statistics Department, University of California. "Arcing the edge". 1997;

Friedman J.H. Greedy Function Approximation: A Gradient Boosting Machine // The Annals of Statistics. 2001. № 5 (29). С. 1189–1232.

Ансамбли моделей: gradient boosting

Реализации градиентного бустинга над решающими деревьями:

XGBoost (в стандартном составе scikit-learn, поддерживается и развивается усилиями сообщества на принципах open source)

1. Chen T., Guestrin C. XGBoost: [A Scalable Tree Boosting System](#). San Francisco, California, USA: ACM Press, 2016. 785–794 с.
2. <https://xgboost.ai/>

LightGBM (первоначально разработана и поддерживалась **Microsoft**, сейчас – community efforts, на принципах open source)

1. Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. "[LightGBM: A Highly Efficient Gradient Boosting Decision Tree](#)". Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3149-3157.
2. Qi Meng, Guolin Ke, Taifeng Wang, Wei Chen, Qiwei Ye, Zhi-Ming Ma, Tie-Yan Liu. "[A Communication-Efficient Parallel Algorithm for Decision Tree](#)". Advances in Neural Information Processing Systems 29 (NIPS 2016), pp. 1279-1287.
3. Huan Zhang, Si Si and Cho-Jui Hsieh. "[GPU Acceleration for Large-scale Tree Boosting](#)". SysML Conference, 2018.
4. <https://github.com/microsoft/LightGBM>

CatBoost (**Яндекс**, развитие и поддержка на принципах open source)

1. Anna Veronika Dorogush, Andrey Gulin, Gleb Gusev, Nikita Kazeev, Liudmila Ostroumova Prokhorenkova, Aleksandr Vorobev "[Fighting biases with dynamic boosting](#)". arXiv:1706.09516, 2017.
2. Anna Veronika Dorogush, Vasily Ershov, Andrey Gulin "[CatBoost: gradient boosting with categorical features support](#)". Workshop on ML Systems at NIPS 2017.
3. <https://catboost.ai/>

Ансамбли моделей: gradient boosting

Особенности алгоритмов бустинга:

- + Наиболее выразительные модели среди всех «классических» методов;
- + Наибольшая точность (в большинстве случаев) среди всех «классических» методов;
- + Наиболее гибкие методы, в практическом/«спортивном» применении (задачи на kaggle.com) чаще всего показывают лучшие результаты среди задач на табличных данных;
- + Можно использовать общий подход, используя другие слабые быстро обучаемые модели
- + Процесс оптимизации параллелизуется, есть ускоренные реализации на GPU, распределенные реализации;

- Склонны к переобучению: могут градиентно «настраиваться» на выбросы/шум, теряя обобщающую способность
- Много гиперпараметров (параметры базовых алгоритмов, параметры градиентной оптимизации, параметры регуляризаций...); настройка гиперпараметров на валидационной выборке может требовать большого количества итераций, что приводит к т.н. «утечке данных» из валидационной выборки в обучение
- Обучение на больших объемах данных может требовать существенных вычислительных ресурсов